後編

システムの記事

FlexRayの実現に向けた時間駆動型通信リアルタイムOSを開発

——ミドルウェアを利用してFlexRay通信の要件を洗い出す

服部博行

制御系の車載ネットワークとして注目されているFlexRay通信だが，実設計時にどのような点を考慮して開発するべきか不明確な部分も多い．そこで，筆者らは要件の洗い出しを行うため，研究目的としてFlexRay通信ミドルウェアを開発した．ここでは，本ミドルウェアの特徴や構成を紹介する．（編集部）

前編（本誌2006年4月号，pp.115-121），中編（同5月号，pp.140-147）では，FlexRay通信が従来のイベント・ドリブン（事象駆動）型でなくタイム・トリガ（時間駆動）型の通信方式をとること，そしてこのタイム・トリガ機能を実現するために新たに筆者ら（ヴィッツ）と名古屋大学 大学院情報科学研究科 高田・冨山研究室が共同で開発したリアルタイムOS「TT-OS」の特徴や機能を説明しました．

本稿では，タイム・トリガ方式のTT-OSと連携して動作するFlexRay通信用ミドルウェアについて解説します．なお，本ミドルウェアは，サニー技研が中心となって開発しました（**図1**）．

1 FlexRayミドルウェア開発の目的

前編でもお話ししたように，車載LANは，通信をまったく利用しない段階（第0段階）から，自動車制御用ネットワークを通じてミッション・クリティカルなサービスを提供する段階（第3段階）までに分類されます注1．FlexRay通信は，この第3段階を目ざしている車載LANであり，ネットワークの障害が生じると自動車の基本機能が損なわれるX-by-wireシステムへの利用が考えられています．

注1：名古屋大学 高田広章氏による車載ネットワークの分類である．

● まずはFlexRayの要件を洗い出す

今回開発したFlexRay通信ミドルウェアは，最初からミッション・クリティカルな用途への適用を目ざしたものではありません．まず，これらの用途に利用する場合にどのような機能や安全性が必要になるのかを検討するための，いわば研究用に開発しました．

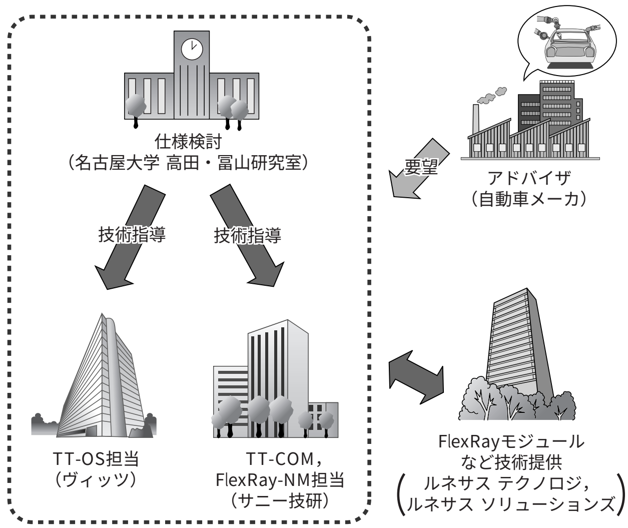


図1　FlexRay関連ソフトウェアの開発体制

本稿で説明しているFlexRay関連のソフトウェアの仕様は，自動車メーカからの要求をもとに，名古屋大学 高田・冨山研究室が中心となって策定した．それをもとに，筆者ら（ヴィッツ）はTOPPERS/OSEKカーネルやタイム・トリガ拡張など，おもにOS部を開発した．また，FlexRay通信ミドルウェア（COM），ネットワーク管理ミドルウェア（NM），ドライバなどの開発はおもにサニー技研が担当した．

KeyWord　イベント・ドリブン，タイム・トリガ，TT-OS，ミドルウェア，TT-COM，FlexRay-NM，FlexRay-DRV，OSEK COM，FT-COM

車載LAN関連の業務に携わっているエンジニアは，FlexRayに大きな期待を寄せています．FlexRayを採用することによって，ECU（電子制御ユニット）間でより多くの情報を速く通知することができ，かつ障害にも強いため，次世代車載LANとして理想的な通信だと思われているようです．しかし，一方では仕様がなかなか確定せず，利用できる環境も存在しないため，何が問題となって，どのような利点が享受されるのか，未知の部分も少なくありません．

例えば，FlexRayの物理層（ハードウェア・レベル）は二重化されており，従来の車載LANと比較して高い信頼性が確保されていることは仕様書から読み取れます[注2]．しかし，こうしたハードウェアである程度確保されている信頼性を，ソフトウェアでも考慮するべきか，また考慮するべきならどのような部位に障害対策を施すべきかなどについて，現時点で正確に規定できる人はほとんどいないと思います．

そこで，これらの要件の洗い出しを行うために本ミドルウェアを開発しました．同時に，FlexRayの問題点やハードウェアの改良点なども検討することを目的としています．また，単にFlexRay通信を利用するだけでは車載LANに求められている機能の洗い出しを行えないと考えました．そこで，現世代の標準的なCAN（controller area network）通信の置き換えとして利用できるミドルウェアとすることにしました．こうすることで，現在の車両システムで利用されている通信を，FlexRayに置き換えてテストを行ったり，必要な機能を抽出することができると考えました．

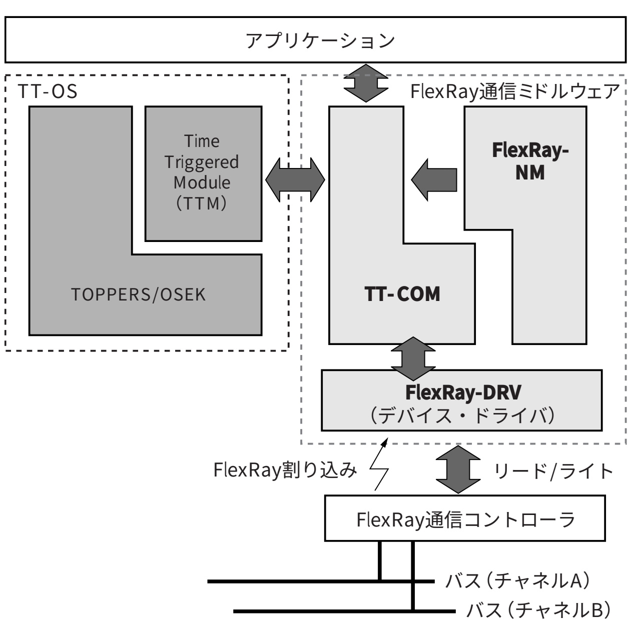


図2　FlexRay通信のためのソフトウェア構成
FlexRay通信ミドルウェアは，OSEK COMと同等の処理を行う通信制御用モジュール（TT-COM）やネットワーク管理用モジュール（FlexRay-NM），デバイス・ドライバ（FlexRay-DRV）からなる．

なお，本ミドルウェアはCAN通信の置き換えを想定しているため，CAN通信と同じレベルの安全性は確保するものの，ミッション・クリティカルな問題に対応可能なフォールト・トレラント（耐故障）機能まではサポートしていません[注3]．つまり，FlexRay通信の最終的な目標であるX-by-wireに対応するための信頼性を確保しているとは言いがたいのです．また，通信速度やソフトウェア・リソースについても，最適なチューニングを施しているわけではありません．

2 FlexRayミドルウェアの構成

今回開発したFlexRay通信のミドルウェアは，次の三つのソフトウェア部品で構成されています（**図2**）．

1) TT-COM（Time Triggered Communication）――アプリケーション・ソフトウェアから直接，データの送受信処理を指令される通信制御用モジュール（OSEK/VDX[注4] OSが利用するOSEK COMと同類の処理を行うミドルウェア）

2) FlexRay-NM（Network Management）――自ノードの状況などをネットワーク経由で他ノードに通知したり，ノードのウェイクアップやスリープなどの管理を行うFlexRay通信のネットワーク管理用モジュール

3) FlexRay-DRV（FlexRay Driver）――FlexRayデバイスを操作するデバイス・ドライバ

● アプリケーションの移植性を向上させるTT-COM

TT-COMは，TT-OSと連携してタイム・トリガ方式による送受信を実行するモジュールです．OSEK/VDX Communication Version 3.0.3（以下，OSEK COMと略す）とOSEK/VDX Fault-Tolerant Communication Version

注2：物理層を二つのチャネル（チャネルA，チャネルB）で接続しており，一方で断線などの故障が起きても，もう一つのチャネルで通信を継続できる．
注3：CAN通信に対する要求事項はすでに整理されているため，その仕様を盛り込むのは容易である．また，今回はCAN通信と純粋に比較できることを目的としている．
注4：OSEK/VDXは，ECU間のインターフェースに関する仕様．OS，通信（COM），ネットワーク管理（NM）について定義している．本仕様を定めている欧州の自動車メーカを中心とした仕様策定団体の名称でもある．

1.0(以下，FT-COMと略す)を参考にして，TT-COMという FlexRay 通信に適した仕様を策定して実装しています.

TT-COMは，OSEK COMやFT-COMにおけるデバイスの隠ぺいや通信 API(application programming interface)の単一化，通信処理に共通するデータのパッキング，プロセッサおよび通信依存のバイト・オーダ自動変換などの思想を継承しています.

TT-COM は，タイム・トリガ型の通信を対象としたFlex Ray 通信専用の仕様です．そのため，イベント・ドリブン型の通信との結合は行いません(FlexRayでは，動的スロットを利用したイベント・ドリブンに似た通信は可能)．将来的には，イベント・ドリブン型の通信仕様であるOSEK COMとの結合を考えていますが，現時点ではタイム・トリガ型の通信の要求を抽出するためにあえて独立に実現しました．そのため，OSEK COMに含まれる内部通信[注5]機能がTT-COMには備わっていません(**図3**).

また，TT-COMはFT-COMが対応しているフォールト・トレラント機能に対応していません．今後，本ミドルウェアをミッション・クリティカルな制御に利用するのであれば，フォールト・トレラント機能に対するなんらかの対策を施す必要があることは理解しています．しかし，この機能部位の参考となる仕様は，現在のところFT-COMしか見当たりません．当然，FT-COMを参考にして仕様の検討を行いましたが，FT-COMのフォールト・トレラント機能をそのまま利用しても意味がないと判断しました.

FT-COM はTTCAN(time triggered CAN)やTTP/C(time triggered protocol/C-class)[注6]などの通信用として策定された仕様だと思われたからです．これらの通信規格ではデバイスの冗長対策は施されていません．そのためソフトウェア，つまりFT-COMの内部で冗長対策を実施しなくてはなりません．一方，FlexRay通信は，前述したように物理層で二重化され，通信経路でも二重化されています．ハードウェアとしての冗長対策が施されているため，ソフトウェアによる対策に意味があるとは思えません(p.110のコラム「FlexRay通信ではソフトウェアによる冗長対策はほんとうに不要？」を参照)．このような理由から，TT-COMではフォールト・トレラント機能に対応していません.

注5：内部通信とは，同一ECUの中で稼働するタスク間，およびタスクと割り込みルーチンの間の情報交換のこと.
注6：TTCANやTTP/Cは，安全性を重視したタイム・トリガ方式のネットワーク規格.

では，FlexRay通信用のソフトウェアにフォールト・トレラント機能は不要でしょうか？ 残念ながら，今のところ正しく判断できるだけの経験や情報を筆者らは持ち合わせていません．FlexRay通信の信頼性を上げるために必要となる対策については，これから本ミドルウェアを利用して見つけていきたいと考えています.

ただ，現段階でわかっていることもあります．それは，ハードウェアによる冗長対策を施した場合，その対策結果をアプリケーションに通知する方法がないということです．少なくとも，ハードウェアを含めたシステムとしてどのような状況になっているかを，アプリケーションに連絡する方法が必要だと思います.

OSEK COMでは，各メッセージの送受信処理において共通となる処理はCOMモジュールに含まれています(6)．TT-COMはこの思想を踏襲し，アプリケーション設計者が同じような処理を何度も設計しなくてもいいように共通機能を実装しています.

例えば，**図4**のようにTT-COMという通信の抽象化ミドルウェアが存在しないと，通信相手もしくはネットワーク上に流れるデータ規則に従ったエンディアンに変換する必要があります(この場合，送信時と受信時)．また，データ属性が類似しているために，アプリケーションにおける

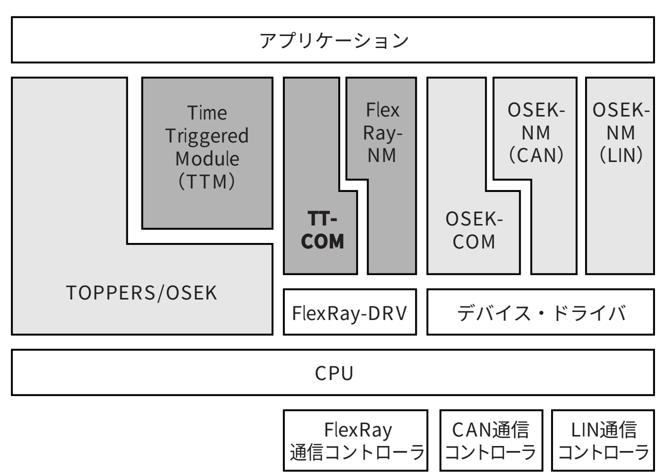


図3　TT-COMと各種COMの関係
FlexRay通信ミドルウェアを利用し，さらにCANやLIN通信を利用したときのモジュールの関係図を示す．OS上にTTMが配置され，タイム・トリガ処理が実現される．FlexRay通信の送受信は，TT-OS(TOPPERS/OSEK＋TTM)，TT-COM，FlexRay-NMにより実現される(図の濃い灰色で示した部分)．内部通信および外部通信(CAN，LIN)は，OSやOSEK-COM，OSEK-NMで実現される．なお，CANとLINではネットワークの管理方法が異なるので，現状では2種類のOSEK-NMが必要となる．将来的には，TT-COMとOSEK-COMをマージして，アプリケーションからすべての通信を単一のCOMで指令することにより実現するのが望ましい.

データ生成(消費)タイミングや生成(消費)位置が異なるデータが同一メッセージにパッキングされて送受信される場合があります．アプリケーション・ソフトウェアの場合，適切な位置で所望するデータのパッキングやアンパッキングを行う必要があり，通常は汎用性のない専用処理となります．しかし，TT-COMモジュールとしてパッキング/アンパッキング処理を含むことで，あらかじめ定められた規則にのっとって処理が行われます．

データによって，現在の状態を通知もしくは取得すればよいものと，データの変化をすべて通知もしくは取得した

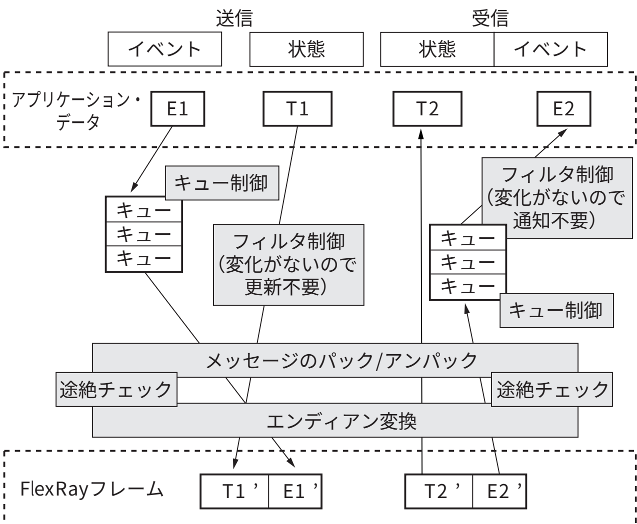


図4　TT-COMがなかったら…

TT-COMがないと，アプリケーションから各種処理を実施する必要がある．図の灰色で示した部分はTT-COMで実行できる．

表1　TT-COMのAPI

TT-COM サービス関数	API	呼び出し元	備　考
アプリケーションからメッセージ領域へのアクセス	ttSendMessage	アプリケーション	メッセージ送信
	ttReceiveMessage	アプリケーション	メッセージ受信
フレーム送信実行処理	ttSendFrame	アプリケーション	メッセージ送信処理(送信コピー処理)
フレーム受信実行処理	ttReceiveFrame	アプリケーション	メッセージ受信処理(受信コピー処理)
TT-COM イニシャル関数	ttInitCOM	スタートアップ・ハンドラ	TT-COMの初期化
TT-COMバッファ初期化関数	ttInitBufferCOM	アプリケーション	TT-COMのバッファ初期化
TT-COM ステータス情報	ttReadFlag	アプリケーション	フラグの読み出し
TT-COM 周期処理関数	ttTickCOM	アプリケーション	途絶判定処理 送信キューにたまったアプリケーション・データの送信処理
FIFO受信フレーム取得	ttReadFifo	アプリケーション	FIFO受信フレームの取得処理

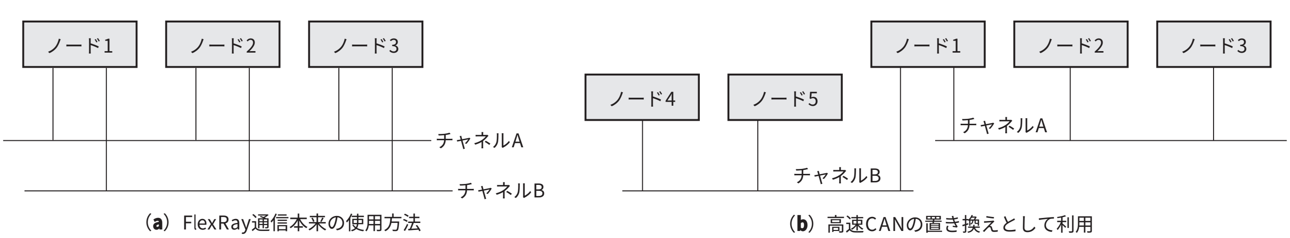
Column　FlexRay通信ではソフトウェアによる冗長対策はほんとうに不要？

現在考えられているFlexRay通信では，通信経路を二重化して利用するため，確かにFT-COMと同等の冗長対策をソフトウェアで行うことは不要だと思います．

しかし，かりにFlexRayの通信経路を二重化して使わず，それぞれの経路で別々のネットワークを構成した場合，ハードウェアで二重化されないので，FT-COMのようなソフトウェアによる冗長対策が必要になるかもしれません(**図A-1**)．

実際，ミッション・クリティカルでない製品分野で高速CANの置き換えとして利用することも考えられます．この場合，FlexRayのチャネルA，チャネルBを別々に利用するので，冗長対策を施していません．制御対象が変われば求められる品質も変わるため，考えかたは一つではありません．

このような考えかたは，ある意味日本的かもしれません．欧米では，高信頼性を得るために冗長化する傾向があるように思います．一方，日本は冗長化しなくても冗長化するのと同じくらいの高信頼性を得るために関連する部署がそれぞれ努力し，協力しています．結果として，冗長化する必要のない有利な設計が可能となります．これもお国柄でしょうか？



図A-1　バスの冗長対策

(a)は，FlexRay通信本来の使用方法．バスを二重化することで安全性を得ている．(b)の場合，CAN同等の信頼性を得られればよいので，バスの二重化といった冗長性を排除してネットワークを構築する．(b)のようなケースでは，ノード1は一種のゲートウェイのような役割を担っている．

いものがあります．こうしたデータの通知/取得も，アプリケーション・ソフトウェアで個別に処理することもできますが，TT-COMモジュール内にあらかじめ備えておくこともできます．

このようにCOMモジュールを利用することで，アプリケーション設計者の作業を軽減するばかりでなく，アプリケーション・ソフトウェアの移植性を向上させることもできます．

● TT-COMには9個のAPIを用意

TT-COMのAPIは，データ送信用の関数と受信用の関数のみであることが理想的でしょう．しかし，実際にはネットワークの状態やエラー発生時の対応手段などを通知するためのAPIなども必要だと考えます．そこで，TT-COMのAPIとして**表1**に示す九つの関数を準備しました．

基本的には，アプリケーションからデータの送受信を行いたい場合，所望するデータを用意し，ttSendMessage，ttReceiveMessageによって送信および受信を指令します．通常，TT-OSと連動してFlexRay通信コントローラから静的に指定された（つまり一定の）タイミングでメッセージの送受信を行います．しかし，アプリケーションで明示的に指令したい場合を想定して，送信フレームを操作するための関数（ttSendFrame，ttReceiveFrame）も用意しました．そのほかの関数としては，TT-COMの初期化関数（通常は自動的に初期化されるが，障害発生後の復旧などのために準備）や，TT-COMの状態を取得するための情報取得用関数に対応しました．

● 各ノードがネットワークを管理するためのFlexRay-NM

FlexRay-NMは，FlexRay通信を実現するために開発したネットワーク管理用のモジュールです．OSEK/VDX Network Management Version 2.5.3を参考にしながら，FlexRayの特徴を取り入れた仕様になっています．

FlexRay仕様では，基本的にはデバイス（FlexRay通信コントローラ）でネットワーク管理機能を実現することになっています．そのため，ミドルウェアにおける機能の作り込みは行っていません（このあたりの詳細は後述）．

ネットワーク管理機能とは，その名のとおり「ネットワークの状況を管理する機能」です．つまり，ネットワークへのノードの参加および離脱，ネットワークに接続されているノード数，自ノードや他ノードの動作状況などを管理（把握）する機能です．こうしたネットワーク状況に関する情報はFlexRayの送信フレーム内に配置されます．具体的には，FlexRay送信用の静的フレームに，ノードのアクティブ情報（ECU制御処理状態）とスリープ可能情報（ECU

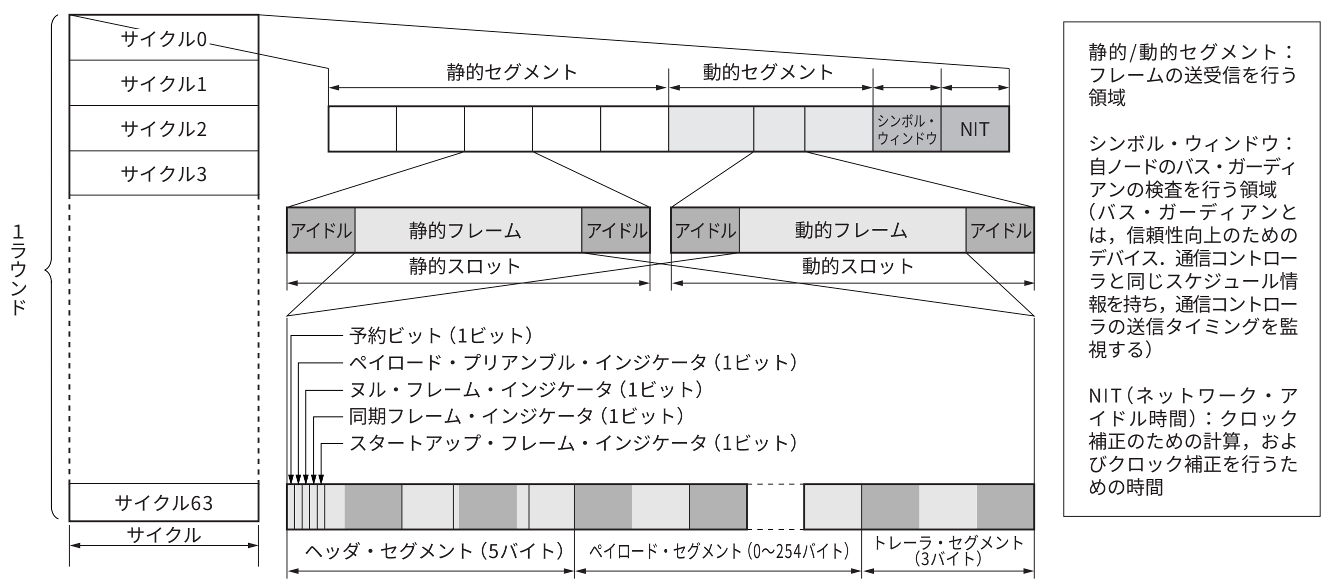


図5 ラウンド，サイクル，フレームの関係図

1ラウンドは最大64サイクルからなる．1サイクルは，静的セグメント，動的セグメント，シンボル・ウィンドウ，NIT（ネットワーク・アイドル時間）で構成される．静的セグメントはシステム設計時に決定した長さのデータ・フレームを扱うため，静的スロット長は固定．一方，動的セグメントは可変長のデータ・フレームを扱うため，動的スロット長も可変である．フレーム・フォーマットは静的フレーム，動的フレームともに，ヘッダ，ペイロード，トレーラという三つのセグメントで構成される．今回開発したFlexRay-NMは，ペイロード上にNM（ネットワーク管理）ベクタを搭載してネットワーク管理を行う．NMベクタは，ペイロード・セグメントの最初の0～12バイトを割り当てられ，ユーザが定義するネットワーク管理を行う．静的セグメントで利用される．

表2　FlexRay-NMのAPI

関数名	呼び出し元	説　明
frInitNM(void)	スタートアップ・ハンドラ	FlexRay-NMモジュールの初期化関数
frTickNM(void)	アプリケーション	NM周期処理関数
frSetSleepRequest(frSleepRequestType)	アプリケーション	スリープ要求の通知関数
frGetStatus(frNMStatusType *)	アプリケーション	NMステータス取得関数
frGetErrorStatus(frErrorStatusType *)	アプリケーション	通信異常要因の取得関数
frExitStopMode(void)	アプリケーション	休止モードの解除関数
frGetNodeMembership(frNodeIdType, frNodeStatusType *)	アプリケーション	指定ノード情報の取得関数
frGetConfig(frNodeStatusType **)	アプリケーション	全ノード情報の取得関数

を休止できるかどうか)に応じたビットを設け，各ノードが自ノードの状況をネットワークに通知します(**図5**)．この情報をもとに，ノードどうしでお互いの接続状況を把握することで，ネットワークをノードごとに管理します．つまり，あるノードが総合的にネットワークを管理するのではなく，各ノードが独自にネットワークの状況を監視するわけです．

上述したように，各ノードは静的フレームにネットワーク管理情報を配置して，フレームの送信時にノード状況をネットワークに通知します．FlexRayの仕様では，このほとんどの処理をFlexRay通信コントローラで行いますが，ネットワークに接続されるすべてのノード数分のビット情報を用意する必要があり，かならずしも効率の良い管理方法とは言えません．

なお，筆者らが今回FlexRay-NMの仕様を検討したとき，静的フレーム内に用意する管理ビットは自ノードのみの情報でも管理できるようにしたり，ネットワーク管理専用のメッセージを作成する方法が望ましいのではないかといった議論も行いました．しかし，仕様で決まっている以上，それと異なる方法を実装しても仕様のバージョンアップに対応しづらいため，上記のような方法をとっています．

● ネットワーク状況をアプリケーションに伝えるAPI

FlexRay-NMのAPIは，ネットワーク状況の情報収集用と自ノードの状況設定用[注7]の関数で構成されています．

ネットワーク管理用のモジュールなので，ネットワークの現在の状況をアプリケーションに通知する必要があります．そのため，FlexRay-NMのAPIは，アプリケーションからの情報収集要求に対応するため，情報収集用関数を用意しています(**表2**)．この関数のうち，frGetStatus，frGetErrorStatusはネットワーク管理モジュール(FlexRay-NM)の状況を，frGetNodeMembershipとfrGetConfigはネットワークの状況をアプリケーションに通知するための関数です．

また，ネットワーク全体を管理するうえで自ノードの状況を他ノードに通知する機能も必要になります．通常，ノードの状況はアプリケーションで管理されています．例えば，自ノードの処理が終了したり，休止モードに移行できる状況にあるかどうかなどはアプリケーションで判断しています．自ノードの状況を自ノードのFlexRay-NMに通知し，そこからネットワーク全体に通知することとなります．これを実現するため，frSetSleepRequestやfrExitStopModeなどの関数を用意しています．

frSetSleepRequestは，自ノードがスリープ状態に移行できる状況が整った，もしくはスリープ状態に移行したいときに用いる関数です．またこの関数は，ネットワークを停止させるための操作でも利用します．frExitStopModeは，自ノードが異常状態から復旧し，正常動作が可能となったことを全ノードに通知する関数ですが，この関数もネットワーク全体で統一された復旧手続きを提供しています．

● FlexRay-DRVからTTMを介してタイミングをとる

FlexRay-DRVは，ルネサス テクノロジの通信コントローラを利用して，FlexRay通信を実現するデバイス・ドライバです．本ドライバの開発段階では，同社からFlexRay通信コントローラを搭載したマイコンが発売されていなかったため，評価ボード(M3A-0847G21)を用いました(**写真1**)．このボードには，同社の32ビット・マイコン「M32C/85」が搭載されています．ボード裏面に搭載されているFPGAにFlexRay通信コントローラ機能が実装されています．このFlexRay通信コントローラには，ドイツ

注7：状況の通知とネットワーク全体の統一的な手続きを用意して，ネットワーク全体の管理を可能とする機能を提供する．

Bosch社のIPコアを利用しています．Bosch社のFlexRay通信コントローラIPコアは国内の半導体メーカの多くが採用を予定しているため，ルネサス テクノロジ以外のコントローラについても比較的容易にドライバを移植できると考えています．

FlexRay-DRVについて，最終的にはソース・コードを公開する予定です．ただし，当分の間はルネサス テクノロジ製の評価ボード（もしくは現在開発中の同社のFlexRay通信コントローラ搭載マイコン）で利用できるバイナリ・コードを提供することになります[注8]．

本デバイス・ドライバは，TTM（Time Triggered Module；詳細は本稿の前編，中編を参照）と関連しています．TTMは，FlexRay通信を実現するためのタイム・トリガ型OSの構成部品であり，いわばタイム・トリガ型OSの中核部品に相当します．デバイス・ドライバとOS中核部品が関係するのは，いささか気味が悪いところがあります．しかし，正確なネットワーク時間を取得することや時間補正の簡略化などを考えると，デバイス・ドライバから直接タイミングを通知したほうがよいと判断しました．

TTMは，FlexRay通信コントローラに内蔵されている，ネットワーク時間で稼働するタイマを利用します．そこで，本デバイス・ドライバではTTMからFlexRay通信コントローラのタイマを操作できる関数を用意しました．公開関数レベルでの結合であるため，例えばFlexRay通信コントローラの変更時やFlexRay以外の通信デバイスを利用したとしても，同等の機能を提供すればタイム・トリガ機能を実現できます．このような関数を利用してTTMを操作することで，TTMからFlexRay通信コントローラに対して所望するサイクル数およびスロット数でのタイマ割り込み指令を設定することができます．また，FlexRay通信コントローラの上記タイマ割り込みは，OSEK OSの管理割り込みに対してTTMを割り込みハンドラとして割り付けることにより，タイム・トリガのタイミングを正確に処理することができます．

本デバイス・ドライバの大きな特徴としてこのようなTTMとの結合が挙げられますが，それ以外は一般的なデバイス・ドライバの機能（デバイスの初期化，ネットワークへのデータ送信，ネットワークからのデータ受信など）を提供しています．

FlexRay-DRVの公開処理関数は，APIとして公開する必要はないと考えています．

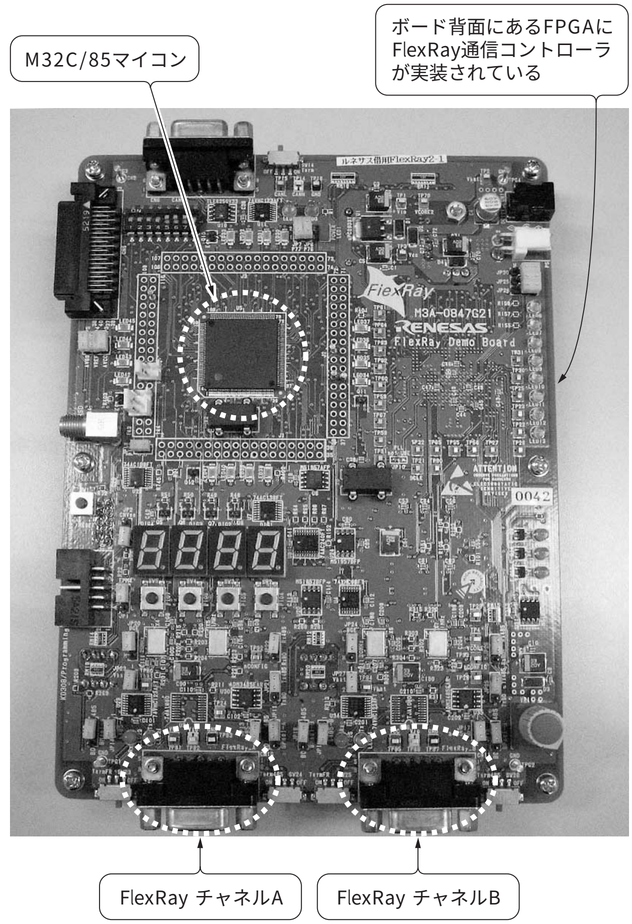


写真1　FlexRay評価ボード

写真はルネサス テクノロジ製のFlexRay評価ボード．通信コントローラはFPGAに実装されている．通信コントローラのIPコアとして，Bosch社のものを利用している．

FlexRay通信を利用する場合，アプリケーションからの指令はTT-COMを経由して送受信処理を行います．また，ネットワークの状況などは，FlexRay-NMを経由して情報の取得を行います．さらに，タイム・トリガ型の処理はTT-OSが静的に作成されたスケジュール・テーブルに従って処理を実行するため，FlexRay-DRVをアプリケーションから直接操作することはほとんどありません（FlexRay-DRVの仕様上，アプリケーションからの呼び出しを考慮した関数は用意していない）．そのため，デバイス・ドライバの関数を説明することは，アプリケーションからの利用を促進してしまうため，ここでの説明は割愛します．

注8：ソース・コードを公開すると，FlexRay通信コントローラの仕様も公開されることになる．そのため，開発中のマイコンが発売されるまでは，本デバイス・ドライバはバイナリ・コードで供給することになる．

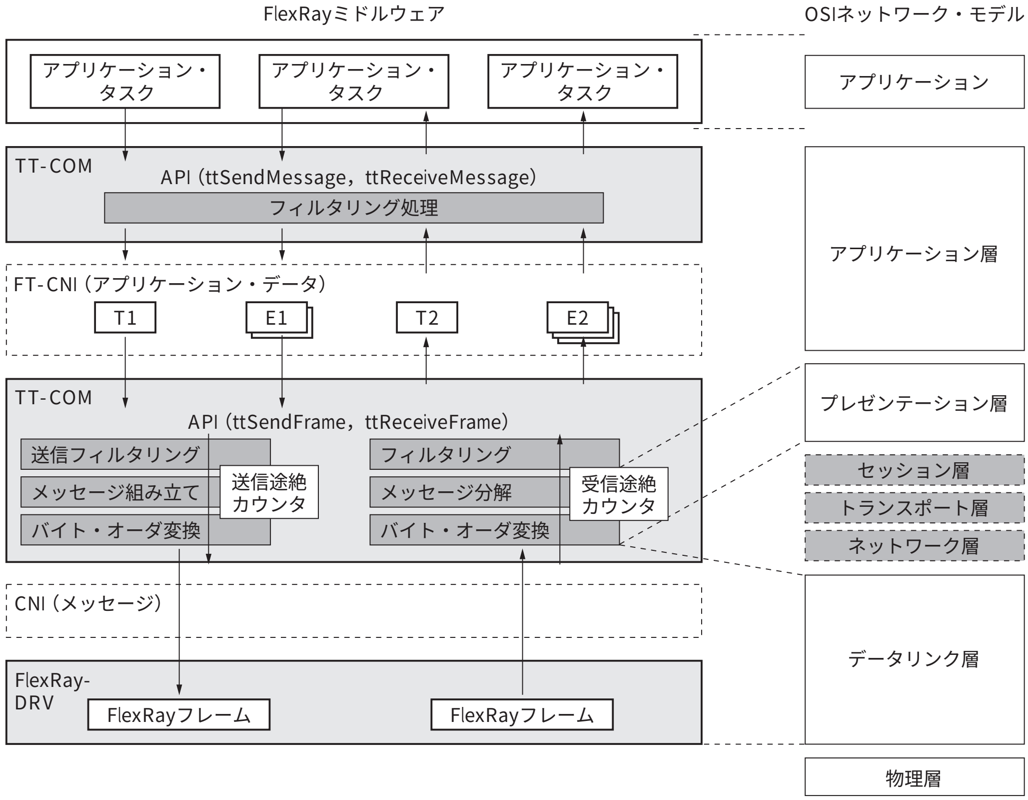


図6　OSIネットワーク・モデルとFlexRayミドルウェア
ソフトウェアも階層構造になっており，OSIのネットワーク・モデルに類似したソフトウェア構成をとっている．

● OS，ミドルウェアともにオープン・ソースとして公開

本ミドルウェアは，研究用途としての利用を目的として開発しました．CAN通信の代替としての機能はすべて実現し，アプリケーションからの利用を抽象化する通信モジュール（TT-COM）やネットワーク管理（FlexRay-NM）を実現し，ボディ系用途で必須となるスリープ/ウェイクアップ機能に対応しています．また，ソフトウェアも階層構造になっており，OSIのネットワーク・モデルに類似したソフトウェア構成になっています（**図6**）．

このように，今回開発したFlexRayミドルウェアはまずCAN通信の代替としてFlexRay通信を利用することを念頭に置き，FlexRay通信への要求を整理することを目的としたため，速度やコード効率などには問題が残っています．

なお，本稿で紹介したFlexRay通信ミドルウェアはすべて（OS，ミドルウェアともに）オープン・ソースとしてTOPPERSプロジェクトから公開されます．これらのソフトウェアは2006年3月末に同プロジェクト会員に対して早期リリースされました．

参考・引用*文献

(1) OSEK/VDX；OSEK/VDX Communication Version 3.0.3，July 20th 2004.
(2) OSEK/VDX；OSEK/VDX Fault-Tolerant Communication Version 1.0，July 24th 2001.
(3) OSEK/VDX；OSEK/VDX Network Management Version 2.5.3，26th July 2004.
(4) 大熊繁監修；車載LANとその応用，トリケップス，2005年6月.
(5) 高田広章；オープンソースのOSEK仕様OS，Automotive Technology Days 2004 Autumn，2004年11月1日.
(6) 服部博行；車載アプリケーション・ソフトウェアの移植性を高める通信仕様とその実装，Design Wave Magazine，2005年9月号，pp.124-133.

はっとり・ひろゆき
（株）ヴィッツ

<筆者プロフィール>
服部博行．自分の興味のあるものは自分で作りたがるやっかいな性格．日曜大工，ビール作りなどで自分の時間を楽しんでいる．最近は，2人乗りの車でドライブを楽しみ，家族からは白い目で見られているが，本人はまったく気にしていない．